CDRL: B015
3 December 1993

UNISYS

Portable, Reusable, Integrated Software Modules (PRISM) Documentation Library Model Document Release 1.0

Central Archive for Reusable Defense Software (CARDS)

Informal Technical Data

Central Archive for Reusable Defense Software

STARS-VC-B015/000/00
3 December 1993

CDRL: B015
3 December 1993

INFORMAL TECHNICAL REPORT
For The
SOFTWARE TECHNOLOGY FOR ADAPTABLE, RELIABLE SYSTEMS (STARS)

Portable, Reusable, Integrated Software Modules (PRISM)
Documentation Library Model Document Release 1.0
Central Archive for Reusable Defense Software (CARDS)

STARS-VC-B015/000/00
3 December 1993

Contract NO. F19628-93-C-0130
Line Item 0002AB

Prepared for:

Electronic Systems Center
Air Force Material Command, USAF
Hanscom AFB, MA 01731-2816

Prepared by:

Azimuth and
Electronic Warfare Associates, Inc.
under contract to
Unisys Corporation
12010 Sunrise Valley Drive
Reston, VA 22091

CDRL: B015
3 December 1993

INFORMAL TECHNICAL REPORT
For The
SOFTWARE TECHNOLOGY FOR ADAPTABLE, RELIABLE SYSTEMS (STARS)

Portable, Reusable, Integrated Software Modules (PRISM)
Documentation Library Model Document Release 1.0
Central Archive for Reusable Defense Software
(CARDS)

STARS-VC-B015/000/00
3 December 1993

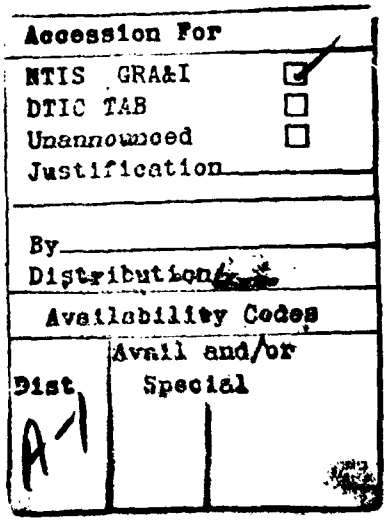
Contract NO. F19628–93–C-0130
Line Item 0002AB

Prepared for:

Electronic Systems Center
Air Force Material Command, USAF
Hanscom AFB, MA 01731–2816

Prepared by:

Azimuth and
Electronic Warfare Associates, Inc.
under contract to
Unisys Corporation
12010 Sunrise Valley Drive
Reston, VA 22091



CDRL: B015
3 December 1993

Data ID: STARS-VC-B015/000/00

Distribution Statement "A"
per Dod Directive 5230.24
Approved for public release, distribution is unlimited

Developed by: Azimuth and Electronic Warfare
Associates, Inc. under contract to Unisys

This document, developed under the Software Technology for Adaptable, Reliable Systems (STARS) Program, is approved for release under Distribution "A" of the Scientific and Technical Information Program Classification Schema (DoD Directive 5230.24) unless otherwise indicated by the U.S. Advanced Research Projects Agency (ARPA) under contract F19628-93-C-0130, the STARS Program is supported by the military services with the U.S. Air Force as the executive contracting agent.

CDRL: B015
3 December 1993

INFORMAL TECHNICAL REPORT
Portable, Reusable, Integrated Software Modules (PRISM)
Documentation Library Model Document Release 1.0
Central Archive for Reusable Defense Software
(CARDS)

Principal Author(s):

Aleisa Petracca *Date*

Les Hayhurst *Date*

George Jackelen *Date*

Approvals:

System Architect: *Kurt Wallnau* *Date*

Program Manager: *Lorraine Martin* *Date*

(Signatures on File)

ABSTRACT

This Portable, Reusable, Integrated Software Modules (PRISM) Documentation Library Release 1.0 Model Document was created by the Central Archive for Reusable Defense Software (CARDS) Program to help disseminate PRISM documentation and knowledge. It represents the current state of the PRISM Documentation Library Model. It is a 'living' document and will be updated with every Library release.

This document describes library modeling and examines modeling concepts and specialization/aggregation hierarchies. This document is specific to PRISM Documentation Library Model Release 1.0 in its description of document types, actual documents available and future direction of this model. The intended audience is anyone desiring an understanding of the PRISM Documentation Library Model and wanting a description of this current Library release. The reader should have an understanding of CARDS and PRISM.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	3 December 1993	Informal Technical Report

4. TITLE AND SUBTITLE
Portable Reusable Integrated Software Modules (PRISM)
Documentation Library Model - Release 1.0
(CARDS)

5. FUNDING NUMBERS
F19628-93-C-0130

6. AUTHOR(S)
Aleisa Petracca
Les Hayhurst
George Jackelen

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Unisys Corporation
12010 Sunrise Valley Drive
Reston, VA 22091

8. PERFORMING ORGANIZATION REPORT NUMBER
STARS-VC-B015/000/00

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Department of the Air Force
Headquarters Electronic Systems Center
Hanscom AFB, MA 01731-5000

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
B015

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Distribution "A"

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Please see Abstract page.

14. SUBJECT TERMS

15. NUMBER OF PAGES
32

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	SAR

CDRL: B015
3 December 1993

Table of Contents

List of Figures

1 Introduction

This document describes the Portable Reusable Integrated Software Modules (PRISM) Documentation Library (PDL) Release 1.0 Model created by the Central Archive for Reusable Defense Software (CARDS) Program. The PDL is an encoding of all available PRISM documents, the documents' hierarchy, and chronological release order. The PDL will be enhanced to reflect new information, new documentation, and new releases of existing documentation. Releases of the PDL will include updates to this document.

The purpose of this Document is to:

- Describe the PDL.
- Describe how the PDL model is related to the CARDS Command Center Library (CCL) model, to CARDS, and to related PRISM projects.
- Provide the reader with an understanding of modeling concepts used to represent the PDL.

Section 2 provides a description of library modeling, specifically the Reuse Library Framework's (RLF's) capabilities and functions, which may be skipped if you are familiar with the topic.

Section 3 discusses the scope, background, history, and model structure of the PDL. How the PDL graphically appears to the user is shown.

Appendix A is a bibliography of sources used to compile this Document.

Appendix B is a glossary of terms and acronyms used within this Document.

Appendix C is a list of actions contained within the PDL.

Terms having a specific technical meaning are **bold** the first time they are used and appear in the glossary. Names of elements of a model (such as concepts, relationships and inferencers) appear in ***bold italic***. Some emphasized words appear in *italic*.

1.1 CARDS Library System History

The current primary source for information about command centers is PRISM. CARDS relies on PRISM for command center models, evolution information, command center domain components, and command center domain documentation. The CARDS library mission is to establish a model-based library of command center components and information, to acquire commercial-off-the-shelf (COTS) and Government-off-the-shelf (GOTS) components, and to qualify the COTS and GOTS for inclusion into the command center libraries.

CARDS has created two libraries (which will be updated, as required, to include new information):

1. The CCL is a model-based library of command center components and information. COTS and GOTS components are acquired and qualified for inclusion in the command center domain. CARDS obtains command center architecture from the PRISM models and components; and in turn, PRISM uses the CARDS CCL to support and further prove the feasibility of integrating software components for use in the command center domain.

2. The PDL includes PRISM development documentation and demonstrations. The PDL was developed to support both CARDS and PRISM. PRISM documentation includes software reuse process information which is domain independent. Therefore, the PDL warrants its own model apart from the CCL.

Since the CCL and PDL are so closely related, some of the entities in the PDL may cross over into the CCL, where necessary. One entity that does cross over from the PDL into the CCL is the Product Assessment Report (PAR); thus PARs appear in both the CCL and PDL. Therefore, there are two independent, but command center related, libraries in the CARDS System Library: the CCL and PDL.

1.2 Summary History of Changes

Since this is the first release of the PDL, there have been no changes to any previous release.

2 Domain Engineering and Library Modeling

CARDS views a library as a library model and a set of applications. The acquired knowledge can be fully represented when harnessed by a model in some formalism. The RLF is the mechanism being used to implement models. RLF integrates a knowledge representation scheme, rule-based inferencing and a graphical browser. The description of the nature of the model requires an overview of the encoding mechanisms.

Although reuse may be approached in a variety of informal ways, CARDS position is that a formal, systematic integration of reuse into the conventional software development process yields substantially greater rewards. The basis for this increased formality is the emerging disciplines of domain analysis and ***domain engineering***. Figure 2-1 illustrates the correspondence relationships between aspects of domain engineering and software engineering. The current state of research and practice of domain analysis and domain engineering are well represented in an IEEE tutorial, Domain Analysis and Software Systems Modeling [PRIE92]. The key points are:

- Domain engineering targets a well-defined application area (or ***domain***) and results in the creation of various products characterizing this domain (e.g., the ***domain model***).

- A key step in domain engineering is domain analysis, which is roughly analogous to software-engineering requirements analysis except that domain analysis describes the requirements of a family of systems (i.e., the requirements of the domain), while software-engineering requirements analysis focuses on the needs of a particular system in question.

A domain model may encompass not just the results of the domain analysis (as just defined), but in addition include other aspects of the domain as well, including a generic architecture for systems within the domain, and reusable components satisfying the conditions of the generic architecture.

This itemization is not meant as a precise and fixed definition of domain engineering (since this is a still-emerging discipline), but rather as a basis for understanding the relationship of CARDS libraries to both domain engineering and software engineering processes.[1]

Note that in the above summary the terms domain analysis, domain model and generic architecture are used, but the terms ***library model*** and ***library modeling*** are conspicuously absent. The reason for distinguishing library models and modeling from domain models and modeling is twofold:

[1] For example, it is not universally accepted that the purpose of domain analysis is merely domain oriented requirements analysis.

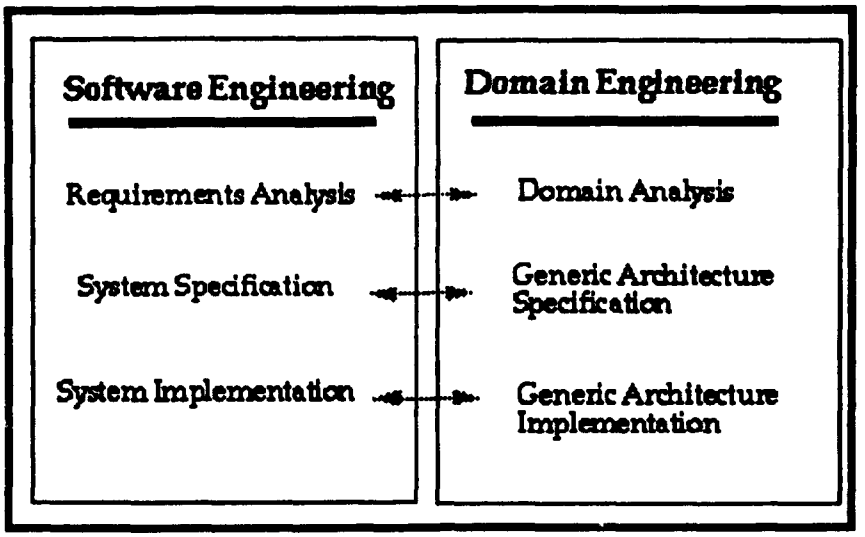


Figure 2-1 Software and Domain Engineering Processes

- Since CARDS is meant to provide a generic capability for constructing domain-specific reuse libraries [STARS93b], it is important that CARDS does not preemptively select one set of domain engineering techniques at the expense of others. Since different application domains (and different Department of Defense (DoD) programs) may be best served by different domain analysis and domain modeling techniques, any such CARDS selection would be premature and counterproductive.

- The decision of which domain engineering by-products to capture, and how to represent them, is a design decision based not only upon the nature of the domain and the domain engineering analysis techniques used, but also on the anticipated use of these products during software engineering. That is, the reuse repository acts as an integrating agent between domain engineering and software engineering processes. The form this integrating agent (i.e., the repository) needs to take is dependent upon both endpoints of the integration relation - domain engineering and software engineering. This point is elaborated in the following section.

2.1 Scoping CARDS Repositories

It is possible, as in Figure 2-2, to scope the repository to capture only the reusable components produced as a result of the implementation phase of domain engineering processes. In Figure 2-2 this scoping is denoted as a parts library. The utility of parts libraries without including some form of architectural context may seem limited, but can be justified in cases where the domain architecture is unstable (i.e., still undergoing technological evolution or standardization), or where a relatively small number of components in the library would not justify the investment in maintaining an up-to-date architectural description. Note in Figure 2-2 that reusable parts can

be 'used' during system implementation, but that an 'understanding' of what parts are available in a repository can aid in system specification.

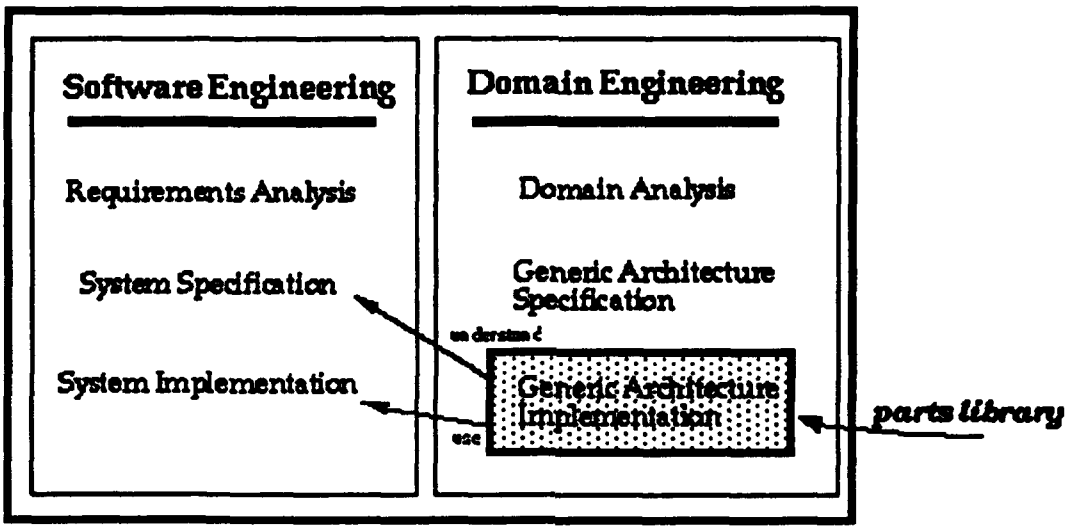


Figure 2-2 Implementation Based Reuse

In Figure 2-3 the scope of the repository has been extended to encompass the reusable components as well as an architectural model capturing the relationships among the components, and describes the purpose of the components within an overall system. One advantage of such a library, denoted as generic architecture library in Figure 2-3, is that the additional context information provided for reusable components can support the automatic composition of systems or parts of systems. For example, if a component implementing part of a database subsystem was selected for use in an application, the generic architecture would provide the basis for the automatic selection and retrieval of any components implied by the selection of the original component (e.g., SQL interface code peculiar to a particular relational database vendor).

In addition, a generic architecture library can make the browsing and searching of large libraries simpler and more meaningful to engineers than, for example, faceted classification schemes requiring strict usage of library-specific keywords. Note that in Figure 2-3 the addition of generic architecture information in the library model extends the 'use' relation to include both system specification and implementation, while requirements analysis can be aided by 'understanding' the architecture supporting applications within a domain.

Figure 2-4 extends the scope of the repository to encompass all of the by-products of domain engineering. This additional scoping broadens the focus of the reuse library to incorporate domain variance as well as domain commonality. While generic architectures focus on what is the same across applications in a domain, a domain model also captures differences across applications in a domain. This broadened focus can support the capture of design rationale for alternative designs of the underlying generic architecture; this, in turn, would be instrumental for evolving the domain architecture in response to new technology and new demands on the previously developed applications.

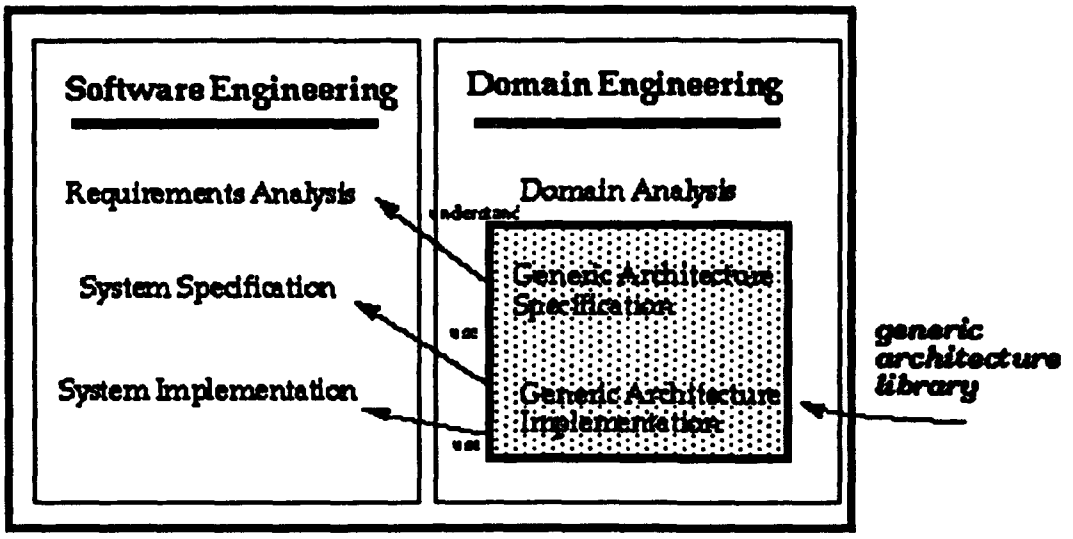


Figure 2-3 Architecture Based Reuse

In Figure 2-4 the advantages of a domain model library are shown both by illustrating the 'use' relationships present throughout software engineering processes, and by illustrating the potential feedback loop from software engineering to the domain model. This is done by capturing the variations of each successive system developed from the reuse library and why these systems varied.

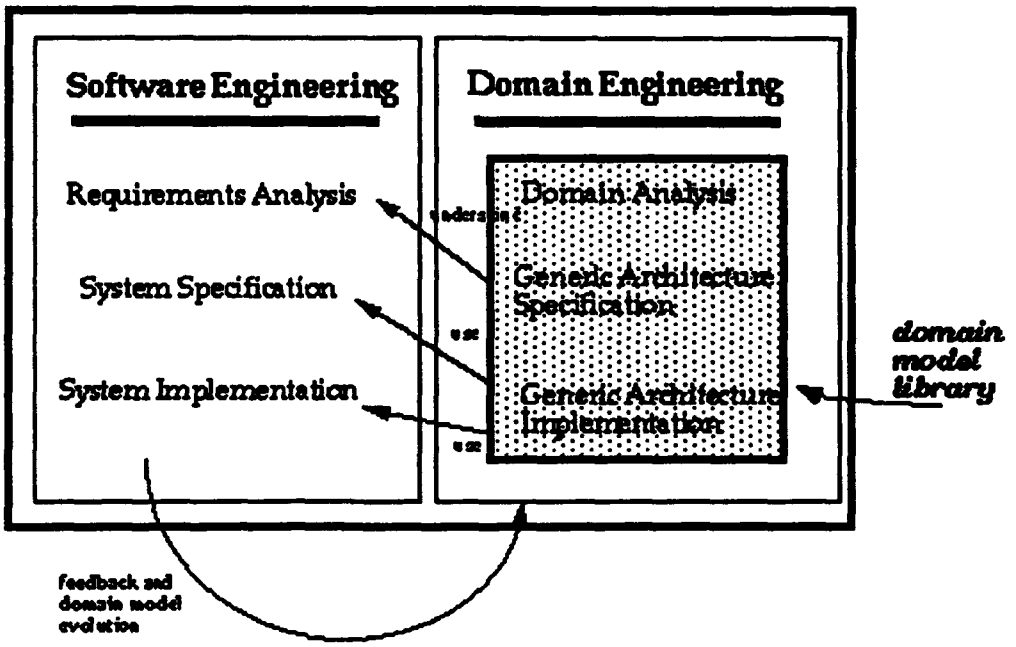


Figure 2-4 Domain Based Reuse

2.2 CARDS Methodology

CARDS views a library as a library model and a set of applications, and the construction of a library model as a design activity balancing various requirements. What goes into and what comes out of a library is dependent upon the library modeling formalism used, the kind of domain analysis conducted, and the kind of library applications needing to be constructed t support the anticipated system engineering processes. The library model includes informatio in addition to that derived from, or pertinent to domain analysis [STARS93b].

There are two approaches to implementing a reuse library:

- ***Component-based*** libraries are organized around a collection of reusable components. The underlying operational concept is that of search and retrieval of individual components. Components found in such libraries are classified in broad, generalized categories. Model-based libraries use domain models as a foundation for library organization and a framework for supporting applications exploiting these models to automate various library services.

- ***Model-based*** libraries encompass information such as domain knowledge, generic architecture specifications, requirements, and implementation restrictions, as well as software artifacts [STARS93b].

One of the visible efforts of library analysis is that of organizing the storage and retrieval of the reusable components. CARDS approach is fundamentally model-based (the motivation has more to do with the retrieval of components, and with the capture of reusable information that is lost in component-based libraries, than with storage). The idea is that one of the major obstacles to reuse is the difficulty potential reusers have in locating reusable products. Models are a very powerful mechanism for organizing components and facilitating user access to them.

A CARDS model is a representation of a specific type of application area (often referred to as a ***domain***) which is simply a limited subject area. Because it mirrors the organization of the part of the real world using the components being stored, it provides a means of organizing and accessing a store mechanism. Components are 'attached' to the model in the spots representing their use in real world applications. The fact that it mirrors the application area makes it a natural way for the user to locate the components they are interested in. This user friendliness is one of the main motivations for organizing the repository storage and retrieval around the model.

The other motivation is that the model is the starting point for potentially many powerful tools to aid the user in reuse related activities. Because the model is 'computationally accessible' and is a detailed picture of what is involved in the domain, it is an excellent starting point for building tools that can 'reason' about the various elements involved in the domain, including the specific components being stored in the library. Inference engines are used to analyze the information encoded in the model and apply various types of understanding to produce results such as software configuration.

2.3 AdaKNET

RLF has a knowledge representation scheme, called AdaKNET, facilitating the classification of library components. AdaKNET can be thought of as a graph where the nodes represent general categories or specific objects, and the edges represent relationships between the nodes. Nodes, representing general classes of knowledge, are called **concepts** or **categories**. There are two basic types of relationships, **specialization** ('is-a') and **aggregation** ('has-a'), described below. Two kinds of hierarchies are built with the concepts and relationships: specialization and aggregation hierarchies.

2.3.1 Specialization

There is exactly one specialization hierarchy in any given 'model'.[1] The specialization hierarchy is a way of defining the concepts. It can be thought of as a glossary and can be compared to the need to declare variables in a traditional computer programming language. The top level concept is usually given a very generic name, such as thing or entity, and all other concepts are defined to be specializations of it. This hierarchy is made by asserting that the 'is-a' unidirectional relationship holds between two concepts, i.e., ***less_general_concept*** specializes ***more_general_concept***. Or, ***less_general_concept*** is-a ***more_general_concept***.

As depicted in Figure 2-5, animal specializes thing, mammal specializes animal, and cow specializes mammal. The modeler can build a set of concepts representing the parts of the domain needing to be referenced. In this view, concepts are represented as a single oval and the specialization links are represented as double lined arrows. A concept represents abstract categories of concise things and may also be called a generic concept, a category or a class.

The specialization hierarchy provides the vocabulary for an AdaKNET model. Because every concept in the model is defined in this hierarchy as being a specialization of some other concept, we have a way of understanding the context of any concept encountered in the model.

Within the specialization hierarchy, the top-level node is very generic and becomes more and more specific at each level until finally the leaf nodes may contain specific examples of their parent concepts. A leaf node representing a particular component or elemental piece of information instantiated from its parent concept is known as an **individual** or **object**. It also may be known as an individual concept, or an instance.

Individuation is the relationship between a parent concept and the instantiation (individual) of that parent concept. Individuation lies within the specialization hierarchy and defines an individuation link between a concept and an individual.

[1]Model is a word potentially having a very specific meaning when used in a very narrow context, but is often used somewhat loosely. It is used in a very general way in this document.

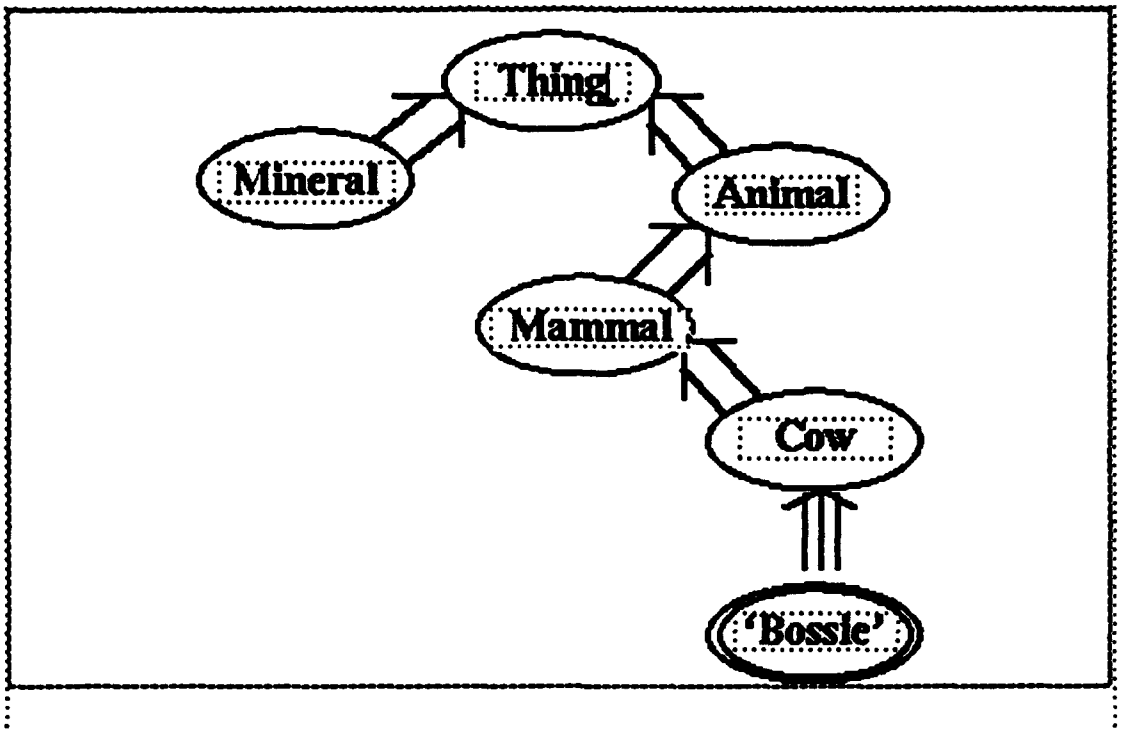


Figure 2-5 Specialization Hierarchy.

Referring to Figure 2-5, note the individual concept called 'Bossie'. In this example, 'Bossie' is our specific *cow*. Yet, 'Bossie' is but one of an infinite number of possibilities for an instantiation of the concept *cow*. As shown in this Figure, individuals are represented as double ovals and the individuation link is represented as a three lined arrow.

2.3.2 Aggregation

Each of the concepts in the specialization hierarchy may have relationships of the 'has-a' nature with any other concept in the model, including itself. These relationships may express attributes, characteristics, features, functional capabilities, requirements or metrics; any relationship existing between two concepts which is not a specialization relationship. Since any concept may have aggregation hierarchy under it, and it is not required that all the units of aggregation structure tie together, the model usually contains many separate pieces of aggregation hierarchy. While every concept must appear in the specialization hierarchy, it is not necessary for every concept to be involved in an aggregation relationship.

The aggregation hierarchy is built by beginning with the concept representing the thing being modeled, such as ***command_center***, and listing all of its 'has-a' relationships. These relationships show substructure, characteristics or other sorts of relationships and are often called **roles**. Each of these relationships has a **name, type** and **range**. The type is the concept being pointed to. The range is an ordered pair, zero to infinity, or some narrower specification, including a **converged**

range such as 2 to 2. This value represents how many copies of that relationship may/must exist simultaneously.

Each concept automatically inherits the aggregation relationships of its ancestors in the specialization hierarchy. AdaKNET supports multiple inheritance; so a concept having more than one parent inherits the aggregation relationships of each. The *range*, and the possible values of the *type*, may be narrowed on subsequent levels of the hierarchy (by the concepts inheriting them) to support the logical structure of the aggregation hierarchy. This is called **role restriction.**

Referring to Figure 2-6, notice that the individual 'Bossie' has what is known as a **filler** satisfying its predecessor's inherited relationship of ***makes_milk*** to the type ***milk***. Individuals must have such aggregational relationships to other individuals. In this case the individual type is 'Bossie's milk' which is an individuation of the concept ***milk***.

In this PDL, aggregation relationships are not represented. To include this representation entails determining how each document is related to other documents, or how each document affects or is affected by other documents. Analysis of this type has currently not been done; however future releases of models may include 'has-a' links.

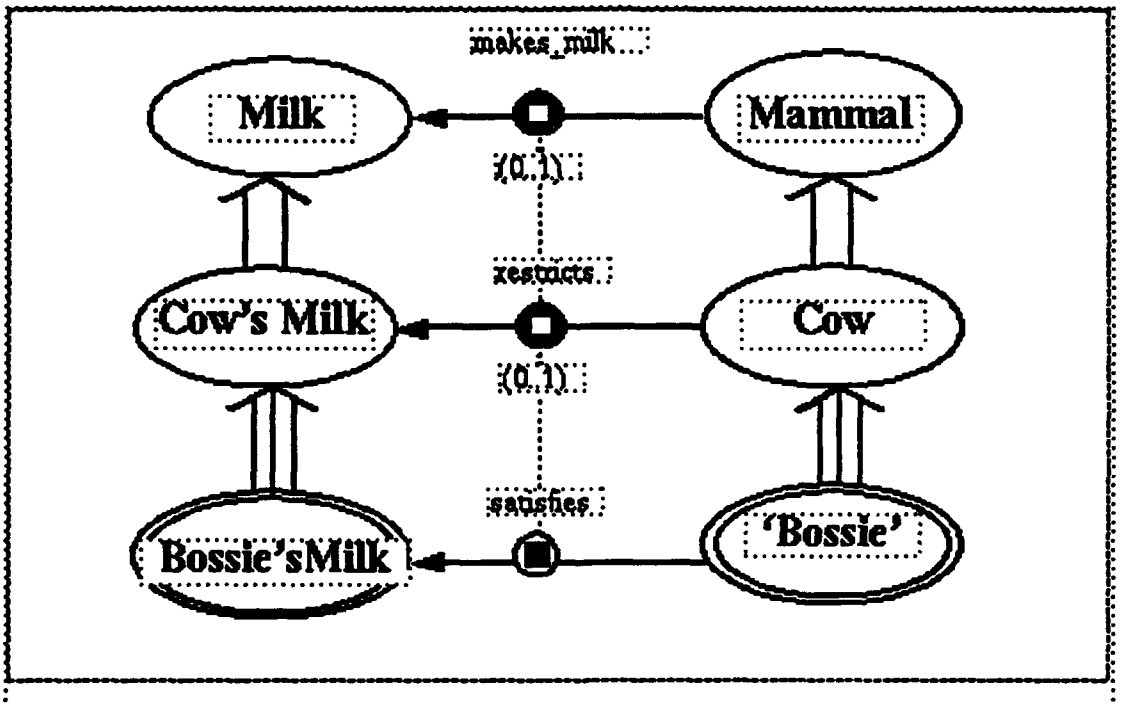


Figure 2-6 Aggregation Hierarchy

2.3.3 The Model

These two types of hierarchy are completely intertwined. In most cases what the modeler and the end user think of as 'the model' is actually a subtree (really a subgraph, but it is generally

thought of as a tree) rooted at the concept representing the thing to be modeled and all the aggregation hierarchy under it.

A full model includes much more than the structural organization expressed in AdaKNET; with the inferencers, discussed below, being the most important other part. However, in this Document the focus is on the structural part of the model. This is because the structure is the aspect that is most usefully described. Discussing why the structure was organized as it was communicates something about the modeler's understanding of the PRISM documentation domain. In contrast, the **inferencers** are aids for accomplishing tasks. Their actual implementation is irrelevant to the end user.

2.3.4 Inferencing

There are two inference engines associated with AdaKNET:

- AdaTAU was written in conjunction with AdaKNET and is a part of the RLF. It is tightly integrated with the Graphical Browser.

- CLIPS (C Language Integrated Production System) [GIAR92] is used in inferencing tools working on the AdaKNET structure. CLIPS was developed for the National Aeronautics and Space Administration (NASA) and is available for a very moderate fee, or free for use on Air Force or NASA projects.

These two inference engines provide similar basic capabilities, but CLIPS is more computationally powerful and also has the capability of querying the AdaKNET model structure.

Both inference engines permit the modeler to write inferencers, units of code expressed as rules about the model. These units, sometimes called **rule-bases**, are associated with specific concepts in the AdaKNET model.

In this PDL, neither inference engine is implemented. In the PRISM documentation domain, there is no apparent necessity for inferencers, since PRISM documents are neither dependent upon each other nor mutually exclusive from each other.

2.3.5 Actions

Any executable program or process, called an 'action', may be associated with any concept by the creator of the library. The mechanism of invoking an 'action' at a concept is typically used to view textual information associated with the concept, but has general applicability to a wide variety of needs.

Actions can be thought of as strings executed in an operating system when the action is invoked. Action 'targets' are the strings representing the object upon which the action acts.

For example, a file describing PRISM process reports may be viewed at the category ***process_report*** by including an action which executes the command:

preview process_report_desc.txt

In this example, 'process_report_desc.txt' is the action target. The same action, with a different action target, may be made available at the category ***architecture_report*** which executes the command:

preview architcture_report_desc.txt

Actions are inherited by concepts in much the same manner as roles. All subconcepts subsumed by a concept declaring an action has that action available. Actions can also be inherited along several specialization links in the case of multiple inheritance.

Actions, again much like roles, can also be restricted at subconcepts below the concept where they are declared. Action targets can be renamed at subconcepts, regardless of inheritance. Appendix C provides more information.

3 PRISM Documentation Library Model

3.1 Scope of the PRISM Documentation Library Model

The PDL Model includes:

- PRISM documents pertinent to the command center domain and architecture (distributions "A" and "C").
- Documents about COTS and GOTS components that were: developed, tested, assessed, certified by PRISM to be qualified by CARDS, and placed within the command center architecture.
- Documents about the PRISM process itself.

3.2 Background

The Generic Command Center (GCC) project (the forerunner of PRISM) integrated components for use in command centers. From the GCC Phase 2 Prototype Summary Report, January 1992 [ESD92]:

> *"The Generic Command Center Phase 2 prototype is an implementation of a portion of the Generic Command Center (GCC) architecture. The purpose of this prototype is to validate the concept of building command centers by integrating large reusable components. The required functionality is that of processing... messages; establishing a database; displaying information in a geographic information system; creating tables of information; and creating briefings that interface with the database. The implementation was consistent with the GCC architecture and used several commercial-off-the-shelf (COTS) products as components.... The results of the GCC Phase 2 are extremely promising toward the concept feasibility of integrating large software components for application in the Command Center domain."*

PRISM has an ambitious goal of fleshing out a real command center generic architecture. It proposes to provide a user with 80% of the required resources to produce a new command center as well as information on acquiring or producing the remainder. The CARDS library models are incrementally encoding and disseminating information generated by PRISM.

3.3 PRISM Documentation Model Access

Upon logging onto AFS and issuing the command "rungb", the Graphical User Interface (also called the library launcher) appears. To access the 1.0:

1. Through the "Choose a Library" pull-down menu, select the library you want, e.g., "PRISM Documentation v1.0"

2. Through the "Choose a Command" pull-down menu, select the "Enter Library" option.

3. Select the "OK" button to enter the desired library.

Release Notes and Help are available and are specific to each library.

3.4 PRISM Documentation Library Model Structure

The documents contained in the PDL are the formal deliverables from contractors working on PRISM and consist mainly of Contract Data Requirements List (CDRL) items and other technical reports delivered to the Government under the contract's Data Accession List. The PRISM documentation was divided into four areas of concentration: Command Center Architecture Reports, Command Center Demonstration Reports, PRISM Process Reports, and Command Center component product assessment reports (see Figure 3-1). These areas of concentration form the first level of PDL specialization detail.

The PDL maintains the same look and feel as the CCL. Upon entering the PDL, the view of the PDL is of the root node, ***prism_documentation***, and its categories and individuals.

ASCII and PostScript versions of the documents contained under this structure are available for extraction; however, only the ASCII versions of the documents may be viewed through the RLF at this time.

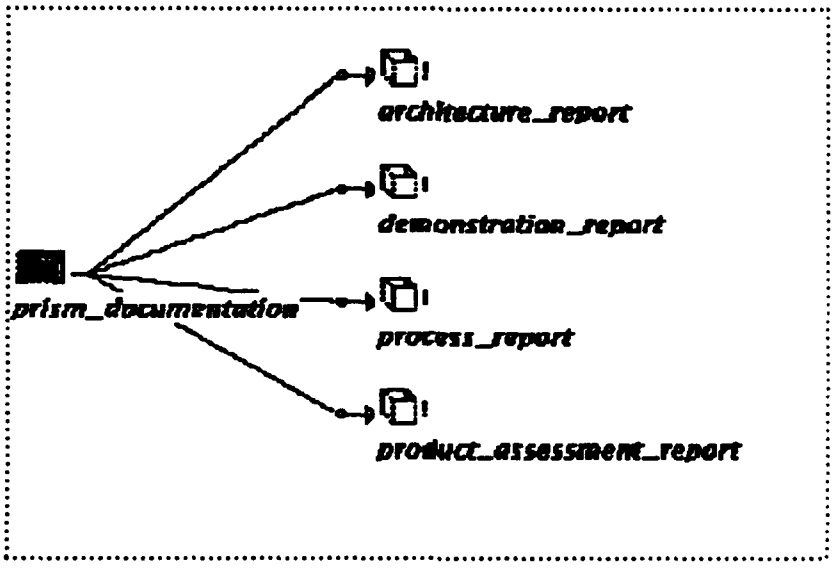


Figure 3-1 PRISM Documentation Model Top-Level View.

3.4.1 Architecture Report

PRISM Architecture Reports (see Figure 3-2) represent the PRISM GCC architecture (GCCA) and serve as an implementation framework for future development of command center systems. The PDL provides the most current version of the Architecture Report.

The purpose of this report is to represent the PRISM GCCA, which is intended to serve as an implementation framework for future development of command center systems. [PRISM93a]

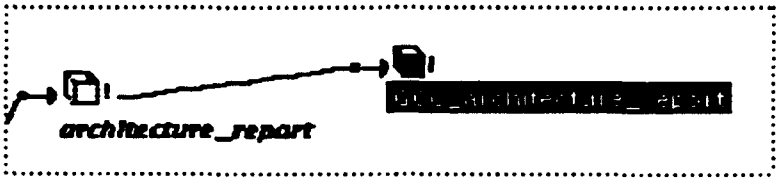


Figure 3-2 Architecture Report and Its Child.

3.4.2 Demonstration Report

Every three months PRISM produces a GCCA demonstration including emphasis on new capabilities being added since the previous quarterly demonstration. A report is generated documenting the environment, results of those demonstrations, and summarizes the PRISM capabilities added to the GCCA. During PRISM's first year, the demonstration reports were called Proof of Concept (POC) Reports. They have since been named Quarterly Demonstration Reports (QDRs). (see Figure 3-3)

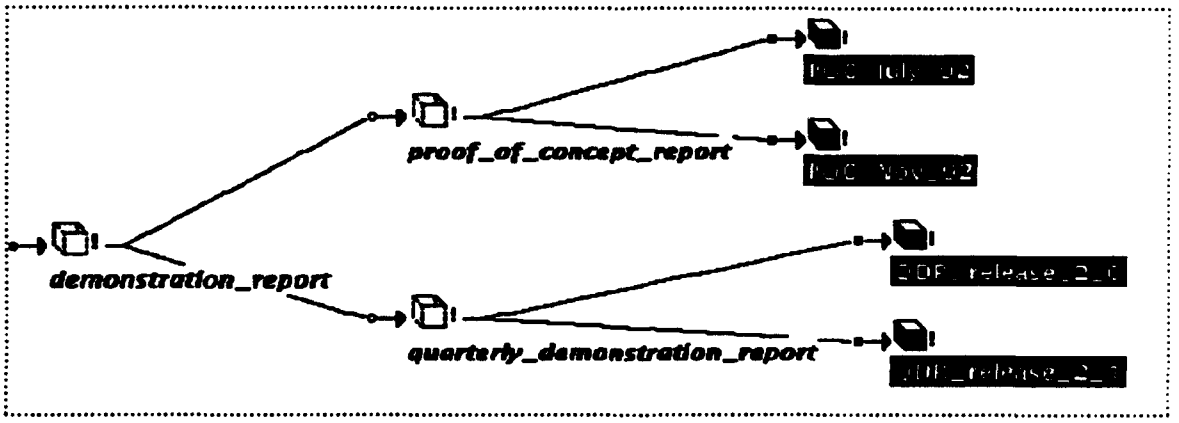


Figure 3-3 Demonstration Report and Its Children.

3.4.2.1 July 1992 Proof of Concept Prototype Report

The July 1992 PRISM POC Prototype Report documents the GCCA requirements and system design. This report also contains the System User's Manual and the Version Description for the GCCA baseline demonstrated in July 1992. This report provides potential GCCA users with descriptions of the software components, design structure, lessons learned, interface definitions, known deficiencies, recommended improvements, and system user instructions. [PRISM93b]

3.4.2.2 November 1992 Proof of Concept Prototype Report

The November 1992 PRISM POC Prototype Report presents a new GCC prototype baseline providing significant increased functionality over previous prototypes. This series of baselines is intended to serve as an implementation framework for command center development; its primary objective is to validate the concept of developing command center systems through the integration of off-the-shelf, reusable software components.

3.4.3 Quarterly Demonstration Report (QDRs)

The PRISM QDRs (Releases 2.0 and 2.1) are generated to document the results of each quarterly demonstration and summarize the capabilities PRISM added to the GCCA.

3.4.4 Process Report

This category of PRISM documentation includes publication of one time CDRLs covering the details of the PRISM process (see Figure 3-4).

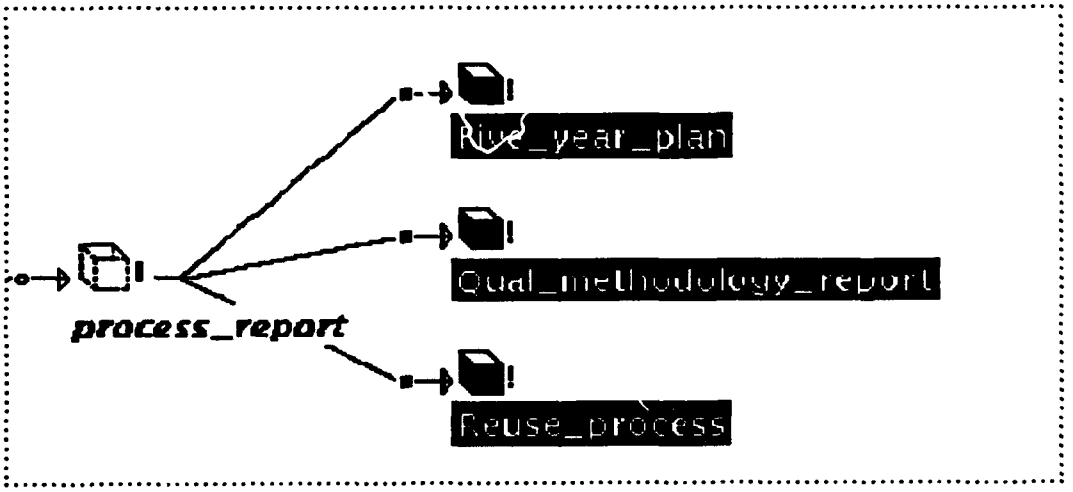


Figure 3-4 Process Report and Its Children.

3.4.4.1 Five Year Plan

PRISM's Five Year Plan is intended to serve as a management tool and assist key staff personnel by identifying and documenting critical program objectives and associated dates and milestones for accomplishing them. The Plan was initially created in March 1992 and has undergone periodic updates and revisions to arrive at this deliverable format. Additionally, this Plan assists in detailed demonstration planning, resource identification and allocation, and tracking future technology trends. [PRISM93f]

3.4.4.2 Qualification Methodology Report

This Report describes a set of procedures and criteria for qualifying software components for incorporation in the generic command center. [PRISM93g]

3.4.4.3 Reusability Process

The PRISM Reusability Process defines a set of procedures and work items for identifying, evaluating, preparing and maintaining reusable software components for PRISM. [PRISM93h]

3.4.5 Product Assessment Report (PARs)

In accordance with the PRISM Qualification Methodology and the PRISM Product Examination Process, PRISM produces PARs (see figure 3-5) on each product included in the quarterly demonstrations. These reports summarize the capabilities of each of the products assessed.

These components were considered for evaluation, but not necessarily at the same level of effort: Operational Support System Joint Automated Message Editing System (OSS-JAMES), Stand-Alone JAMES, Logicon Message Dissemination System (LMDS), Joint Message Analysis and Processing System (JMAPS), Sybase, Oracle, Interbase, Ingres, Informix, DeLorme's XDK, Prior Data Science's InterMAPhics, Rome Laboratory's Common Mapping Toolkit (CMTK), ARC/INFO, DECmessageQ, Trusted Information System's (TIS) Trusted Xenix (with Ethernet LAN interfaces, DoD standard protocols (TCP/IP) and network applications (FTP)), Harris Night Hawk, SEI MTV, UNIX tools lex and yacc, SunNet Manager (SNM) product, SecureWare's Compartmented Mode Workstation (CMW+) for A/UX, Trusted Information System's (TIS) Trusted Xenix B1 platform, AT&T's B1 System V MLS (running on a 3B2), Ultrix MLS+, Verdix Secure LAN (VSLAN) and Boeing MLS LAN.

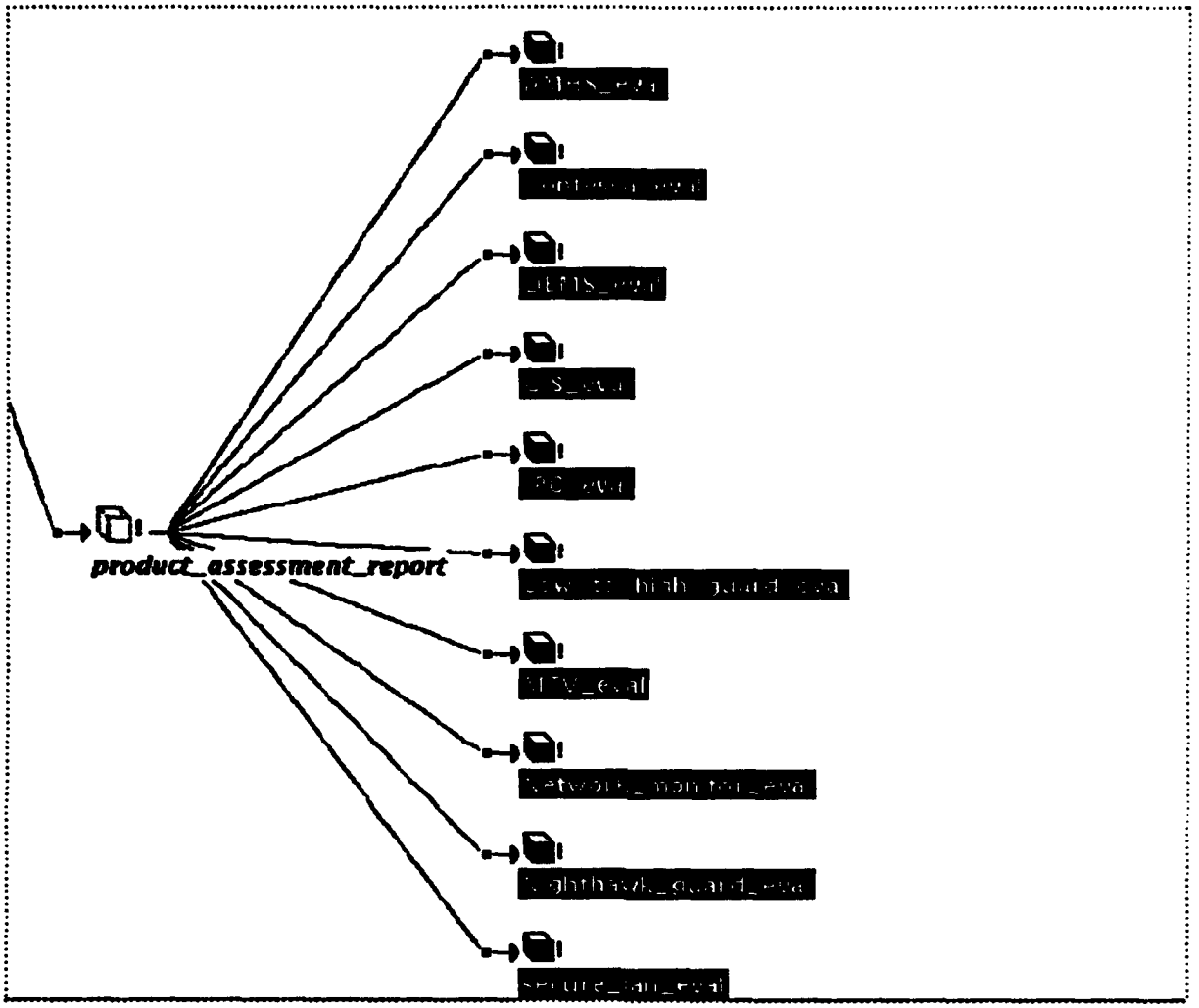


Figure 3-5 Product Assessment Report and Its Children.

3.4.5.1 AMHS Evaluation

This Report describes Automated Message Handling System (AMHS) product assessments performed on products intended for use in the PRISM GCCA, which provides the technologies and components needed to support specific command center objectives. [PRISM93i]

3.4.5.2 Contessa Evaluation

Contexture Systems' Contessa is an application development environment that creates graphical user interface (GUI) based applications which integrate data. [PRISM93j]

3.4.5.3 DBMS Evaluation

This Report documents the products being assessed as candidates for the Database Manager (DBM) component of the PRISM GCCA. The DBM consists of the server and its interface to the databases it controls. This Report includes the Sybase product assessment through integration in the GCC environment and the screening of the Oracle, Interbase, Ingres and Informix products. Sybase met the basic GCCA requirements. [PRISM93k]

3.4.5.4 GIS Evaluation

This Report documents the product assessments of various Geographic Information Systems (GIS). [PRISM93l]

3.4.5.5 Interprocess Communication (IPC) Evaluation

This Report documents the products being assessed as candidates for the IPC component. The IPC provides a common application program interface (API) of communication services across heterogeneous operating systems and networks. [PRISM93m]

3.4.5.6 Low to High Guard Evaluation

This report documents a security mechanism that only allows file transfer of information from one level of security classification to the next higher level of security classification.

3.4.5.7 MTV Evaluation

This Report documents the products being assessed as candidates for the Message Translator and Validator (MTV) component of the PRISM GCCA. The MTV component performs the translation and validation of both character and bit based messages, and the update of the mission database and the mission application displays.

3.4.5.8 Network Monitor Evaluation

The Network Monitor and Control function enables maximum performance and utilization of resources from the IPSs and communication services. [PRISM93p]

3.4.5.9 Night Hawk Guard Evaluation

This Report documents the product assessment of the Low-to-High Message Guard Platform component. [PRISM93q]

3.4.5.10 Secure LAN Evaluation

This Report describes the assessment process of two LAN components: the VSLAN through the integration phase and the Boeing MLS LAN through the screening phase.

3.5 Future Directions/Enhancements

3.5.1 Structural Changes

The main thrust of future development of the PDL will be the addition of PRISM command center component demonstrations using a software tool called ScreenPlay. In these demonstrations, command center software will be displayed as a series of static views ("snap shots") of the actual running of the software.

3.5.2 Action Changes

The addition of system demonstrations using the ScreenPlay utility will warrant the addition of a run_demonstration action to the library. It will be placed under the perform_action menu option appearing after clicking the desired node.

APPENDIX A - Bibliography

[ESD92] Generic Command Center Phase 2 Prototype Summary Report, ESD/AVS Hanscom AFB. Jan 92.

[CARDS] PRISM Document Library User's Guide Release 1.0, STARS-VC-B006/000/00, 3 December 1993.

[GIAR92] CLIPS User's Guide, NASA, Joseph Giarratano, Sept 92.

[PRISM93a] Generic Command Center Architecture Report, Portable, Reusable, Integrated Software Modules (PRISM), Generic Command Center Architecture, Document No. 2Y992-92-1017/5D744-G551354 Revision A, 8 Sept 93.

[PRISM93b] July Proof of Concept Report, Portable, Reusable, Integrated Software Modules (PRISM), Generic Command Center Prototype, Document No. 2Y992-92-1022/5D744-G551355, 22 Aug 93.

[PRISM93c] November Proof of Concept Report, Portable, Reusable, Integrated Software Modules (PRISM), Generic Command Center Prototype, Document No. 2Y992-92-1022/5D744-G551355, 22 Aug 93.

[PRISM93d] Generic Command Center Version 2.0 Quarterly Demonstration Report, Portable, Reusable, Integrated Software Modules (PRISM), Generic Command Center Prototype, Document No. 2Y992-92-1022/5D744-G551355 Revision B, 12 Jul 93.

[PRISM93e] Generic Command Center Version 2.1 Quarterly Demonstration Report, Portable, Reusable, Integrated Software Modules (PRISM), Generic Command Center Prototype, Document No. 2Y992-92-1022/5D744-G551355 Revision C, 12 Jul 93.

[PRISM93f] Five-Year Strategy Plan, Portable, Reusable, Integrated Software Modules (PRISM), Five-Year Strategy, Document No. 2Y992-93-1010/5D744-G551370, 22 Aug 93.

[PRISM93g] Qualification Methodology Report, Portable, Reusable, Integrated Software Modules (PRISM), Qualification Methodology, Document No. 2Y992-92-10xx/5D744-G5513xx, 23 Aug 93.

[PRISM93h] Reusability Process Report, Portable, Reusable, Integrated Software Modules (PRISM), Reusability

Process (Draft), Document No. 2Y992-93-1013/5D744-G551353, 23 Aug 93.

[PRISM93i] Automated Message Handling System (AMHS) Product Assessment Report, Portable, Reusable, Integrated Software Modules (PRISM), File No. PRISM:93:0005, 7 Jul 93.

[PRISM93j] CONTESSA Product Evaluation, Portable, Reusable, Integrated Software Modules (PRISM), File No. PRISM:93:0091, 9 Apr 93.

[PRISM93k] Database Manager System (DBMS) Product Assessment Report, Portable, Reusable, Integrated Software Modules (PRISM), File No. PRISM:93:0006, 7 Jul 93.

[PRISM93l] Geographic Information System (GIS) Product Assessment Report, Portable, Reusable, Integrated Software Modules (PRISM), File No. PRISM:93:0068, 11 May 93.

[PRISM93m] Interprocess Communication (IPC) Component Product Assessment Report, Portable, Reusable, Integrated Software Modules (PRISM), File No. PRISM:93:0088, 7 Jul 93.

[PRISM93n] Low to High Filter Transfer Guard Product Assessment Report, Portable, Reusable, Integrated Software Modules (PRISM), File No. PRISM:93:0007, 7 Jul 93.

[PRISM93o] Message Translator Validator (MTV) Product Assessment Report, Portable, Reusable, Integrated Software Modules (PRISM), File No. PRISM:93:0089, 7 Jul 93.

[PRISM93p] Network Monitor Product Assessment Report, Portable, Reusable, Integrated Software Modules (PRISM), File No. PRISM:93:0093, 14 Jul 93.

[PRISM93q] Low-to-High Message Guard Platform Product Assessment Report, Portable, Reusable, Integrated Software Modules (PRISM), File No. PRISM:93:0090, 7 Jul 93.

[PRISM93r] Secure Local Area Network (LAN) Product Assessment Report, Portable, Reusable, Integrated Software Modules (PRISM), File No. PRISM:93:0094, 16 Jul 93.

[STARS92] RLF Graphical Browser User's Guide, STARS-SC-03065/004A/00, Jan 92.

[STARS93a] RLF User's Manual, Version 4.1, STARS-UC-05156/013/00, Mar 93.

[STARS93b] RLF Modeler's Manual, RLF Version 4.1, STARS-UC-05156/011/00, Feb 93.

[STARS93c] RLF Modeler Tutorial, STARS-UC-05156/020/00, Feb 93.

APPENDIX B - Glossary of Terms and Acronyms

B.1 Terms

action	A mechanism permitting a user of the RLF graphical browser to invoke calls to the underlying operating system, including invoking other tools.
aggregation	Relationship between AdaKNET concepts which express attributes, characteristics, features (as the term is used in FODA), functional capabilities, requirements or metrics; that is, any relationship existing between two concepts which is not a specialization relationship.
application	A system providing a set of general services for solving some type of user problem.
automated message handling	Processing of strictly formatted messages.
category	- see concept.
class	- see concept.
command center	A facility from which a commander and her/his representatives direct operations and control forces. It is organized to gather, process, analyze, display and disseminate planning and operational data and to perform other related tasks.
component	A set of reusable resources related by virtue of being the inputs to various stages of the software design lifecycle, including requirements, design, code, test cases, documentation, etc. Components are the fundamental elements in a reusable software library.
concept	An atomic unit of the AdaKNET knowledge representation scheme, representing an idea or thing, also known as a generic concept, a category or a class.
converged	Said of an AdaKNET role range for which the minimum and maximum have been set to the same value.
domain	An area of activity or knowledge containing applications sharing a set of common capabilities and data.
feature	A prominent or distinctive user-visible aspect, quality, or characteristic of a software system or systems.

filler	A role used only between individuals. The filler of an individual's relationship must adhere to the relationship's restrictions. Both the owner and the 'filler' must be individuals.
generic architecture	A collection of high-level paradigms and constraints characterizing the commonality and variances of the interactions and relationships between the various components in a system.
individual	A knowledge representation of the reusable component of the library. An AdaKNET term for representing a specific instantiation of a concept. Also known as an individual concept, an object or an instance.
individuation	The relationship between an individual to its subsuming concept. Indicates that an individual is an actual instance of the idea represented by the concept.
inferencer (Also called rule-base)	A mechanism using existing facts and rules about the elements of a model to perform reasoning about that model and deduce new facts and rules.
inherit	- see inheritance.
inheritance	A mechanism whereby classes make use of the procedures, attributes, and/or data defined in other classes.
instance	- see individual.
intermediate levels of specialization	Concepts that partition a category into subcategories.
library model	A model representing the domain components and the relationships between them.
library modeling	The process of building a library model that accurately reflects the functionality and structure of the target domain.
model-based library	A library organized around the principle that what matters in a repository is the context in which reusable software components are used and the relationships among components. The focus of a model-based library is the model (requirements, architectures, design decisions and rationales) and the software implementing these models.
multi-level security	Information processing and communications allowing two or more classification levels of information to be

	processed simultaneously within the same system when some users are not cleared for all levels of information present.
(role) name	Refers to the name of an AdaKNET aggregation relationship.
object	- see individual.
(role) range	The number of simultaneous copies consisting of an AdaKNET aggregation relationship.
role	An AdaKNET term referring to an aggregation ('consists of') relationship between two concepts.
role restriction	Refers to an inherited AdaKNET aggregation relationship which is narrowed at the inheriting concept, either by further restricting the range or by further restricting the type.
rule-base (Also called inferencer)	A collection of rules about the elements of a domain. A rule describes the relationships, requirements and constraints among components.
software architecture	High-level paradigms and constraints characterizing the structure of operations and objects, their interfaces and control to support the implementation of applications in a domain. Includes a description of each software component's functionality, name, parameters and their types, and a description of the components' interrelationships.
specialization	The act of declaring that one concept represents a narrowing of the idea represented by another concept.
subsume	Having an "is-a" relation with a concept where the subsuming concept is a larger and more abstract category (e.g., X is_a Z and Y is_a Z therefore Z subsumes X and Y).
(role) type	The allowable range of values of an AdaKNET aggregation role.

B.2 Acronyms

ACC	Air Combat Command
AMHS	Automated Message Handling System

API	Application Interface
ARPA	Advanced Research Projects Agency
CARDS	Central Archive for Reusable Defense Software
CCL	Command Center Library
CDRL	Contract Data Requirements List
COTS	Commercial-Off-The-Shelf
DBM	Database Management
DBMS	Database Management System
DMQ	DECmessageQ
FTP	File Transfer Protocol
GCC	Generic Command Center
GCCA	Generic Command Center Architecture
GIS	Geographic Information System
GOTS	Government-Off-The-Shelf
GUE	Graphical User Interface
IPC	Interprocess Communications
IPS	Information Processing Subsystem
JMAPS	Joint Message Analysis and Processing System
LAN	Local Area Network
LMDS	Logicon Message Dessination System
MLS	Multi-Level Security
MTV	Message Translator Validator
NASA	National Aeronautics and Space Agency
OSS-JAMES	Operational Support System Joint Automated Message Editing System
PAR	Product Assessment Report
PDL	PRISM Documentation Library

POC	Proof Of Concept
PRISM	Portable, Reusable, Integrated Software Modules
QDR	Quarterly Demonstration Report
RLF	Reuse Library Framework
SEI	Software Engineering Institute
SNM	SunNet Manager
STARS	Software Technology for Adaptable, Reliable Systems
TCP/IP	Transmission Communications Protocol/Interface Protocol
TIS	Trusted Information System
VSLAN	Verdix Secure Local Area Network

APPENDIX C - Library Actions

Actions can be any executable command or script. If an 'action' is included at a concept, the phrase *Perform Action* appears on the pop-up menu when the user chooses the node representing that concept. Another pop-up menu appears when *Perform Action* is chosen. This menu displays the action(s) available at that concept. If there are no actions at the concept, the *Perform Action* option is not presented as a menu choice.

The list of invocable actions, a short description, and the concepts at which those actions can currently be called follows:

- **Provide Description** — Displays the Description file for a particular node. The preview utility is used to display the Description file. It is available at:
  - architecture_report
  - demonstration_report
  - process_report
  - product_assessment_report
  - proof_of_concept_report
  - quarterly_demonstration_report
  - prism_documentation
- **View Contents** - Views ASCII text version of the document. The preview utility is used to display the ASCII file, which is in the same directory space as the PostScript file and is extractable:
  - GCC_architecture_report
  - POC_July_92
  - POC_Nov_92
  - QDR_release_2_0
  - QDR_release_2_1
  - Five_year_plan
  - Qual_methodology_report

- Reuse_process
- Secure_lan_eval
- Nighthawk_guard_eval
- Network_monitor_eval
- MTV_eval
- Low_to_high_guard_eval
- IPC_eval
- AMHS_eval
- Contessa_eval
- DBMS_eval
- GIS_eval

- **Extract Contents** - Extracts the contents of directories and/or files from an RLF Library to a user specified target directory. It is available at concepts:
  - GCC_architecture_report
  - POC_July_92
  - POC_Nov_92
  - QDR_release_2_0
  - QDR_release_2_1
  - Five_year_plan
  - Qual_methodology_report
  - Reuse_process
  - Secure_lan_eval
  - Nighthawk_guard_eval

- Network_monitor_eval
- MTV_eval
- Low_to_high_guard_eval
- IPC_eval
- AMHS_eval
- Contessa_eval
- DBMS_eval
- GIS_eval

- **Display Abstract** - Displays an Abstract outlining the features and capabilities of an asset. It is available at concept:
  - GCC_architecture_report
  - POC_July_92
  - POC_Nov_92
  - QDR_release_2_0
  - QDR_release_2_1
  - Five_year_plan
  - Qual_methodology_report
  - Reuse_process
  - Secure_lan_eval
  - Nighthawk_guard_eval
  - Network_monitor_eval
  - MTV_eval
  - Low_to_high_guard_eval

- IPC_eval
- AMHS_eval
- Contessa_eval
- DBMS_eval
- GIS_eval